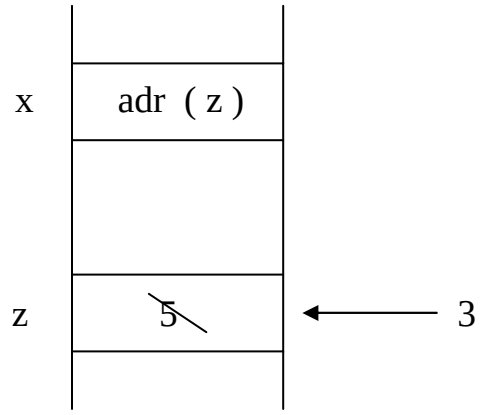
♦ TRANSMITEREA DATELOR CA PARAMETRI.

♦ TRANSMITEREA PRIN ADRESĂ (REFERINȚĂ) .

```
var  z : t ;
- - - - - - - - - - - - - - - - -
procedure  p ( x : t ) ;
- - - - - - - - - - - - - - - - -
begin
      - - - - - -
      x : = 3 ;        ⇐
end ;
- - - - - - - - - - - - - - - - -
z : = 5;
p ( z )
- - - - - - - - - - - - - - - - -
```

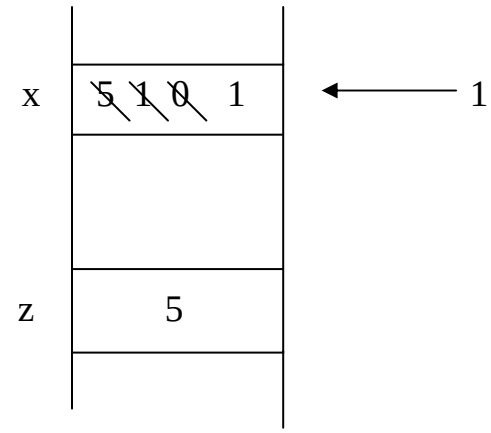


♦ TRANSMITEREA PRIN COPIERE.

♦ A) TRANSMITEREA DE VALOARE :

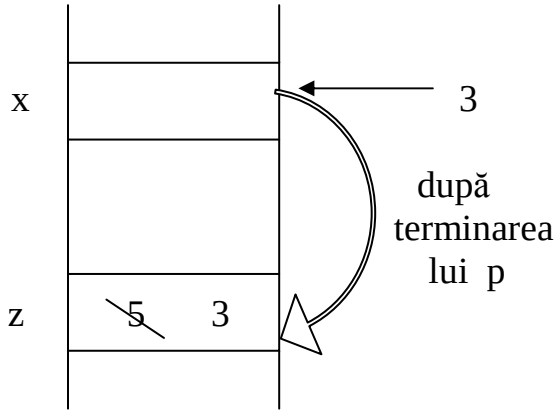
```
var  z : t ;
- - - - - - - - - - - - - - - - -
procedure  p ( x : t ) ;
   var  a : t ;
begin
      a : = x - 1 ;
      - - - - - - - - - - -
      x : = 1 ;          ⇐
end ;

   z : = 5 ;
- - - - - - - - - - - - - - - - -
   p ( z ) ;
   p ( z - 5 ) ;
- - - - - - - - - - - - - - - - -
```



♦ **B) TRANSMITEREA DE REZULTAT :**

```
var  z : t ;
- - - - - - - - - - - - - - - - -
procedure  p ( x : t ) ;
- - - - - - - - - - - - - - - - -
begin
      - - - - - -
      x : = 3 ;
end ;
- - - - - - - - - - - - - - - -
z : = 5 ;
p ( z ) ;
- - - - - - - - - - - - - - - -
```



♦ **C) TRANSMITEREA DE VALOARE ŞI REZULTAT :**

```
var  z : integer ;
- - - - - - - - - - - - - - - - -
procedure  p ( x , y : integer ) ;
- - - - - - - - - - - - - - - - -
begin
    x : = 2 * x ;
    y : = 2 * y ;
end ;

- - - - - - - - - - - - - - - -
z : = 3 ;
p ( z , z ) ;
- - - - - - - - - - - - - - -
```

♦ TRANSMITEREA PRIN NUME .

```
var  x , y , i : integer ;
       t : array [ 1 . . . 100 ] of integer;
- - - - - - - - - - - - - - - - - - - - - - - - - - -
procedure  p ( a , b : integer ) ;
       var  m : integer;
begin
    m : = a ;
    a : = b ;
    b : = m ;
end ;
```

În cazul unui apel p (x , y), secvența executată (considerând că transmiterea parametrilor se face prin nume), va fi :

```
m : = x ;
x : = y ;
y : = m ;
```

Să considerăm următorul apel :

```
i : = 3 ; t [ i ] : = 50 ;
p ( i , t [ i ] ) ;
```

Secvența executată, considerând că transmiterea se face prin nume, va fi :

```
m : = i ;
i : = t [ i ];
t [ i ] : = m;
```

Ca urmare, i = **50** ; t [3] rămâne **50** ; în schimb se modifică t [50], care ia valoarea **3**.

```
var  x : integer ;
- - - - - - - - - - - - - - - - -
procedure  p ( a : integer ) ;
    var  x : integer ;
begin
       x : = 2 ;
       write ( a ) ;   ——————→ aici se scrie 1
       write ( x) ;    ——————→ aici se scrie 2
end ;
- - - - - - - - - - - - - - - - -
x : = 1 ;
p ( x ) ;
- - - - - - - - - - - - - - - - -
```

♦ TRANSMITEREA SUBPROGRAMELOR CA PARAMETRI.

```
type  fnt =  function (x :  integer) : real ;

procedure  tab  (f : fnt ; j , s :  integer) ;
        var  a :  integer ;

begin
        for  a : =  j  to  s  do
              writeln ( a,  f ( a ) ) ;

end ;

{$ F +}

function  f1 ( x : integer ) : real ;

begin
        f1 : = 2 * 3.14 * x

end ;

function  fact  ( x : integer ) : real ;
        var  f : real ;  i  : integer ;

begin
        f : =  1 ;
        for  i  : =  1  to   x   do
                    f : = f  *  i  ;
        fact  : = f

end ;

{$ F -}

- - - - - - - - - - - - - - - - - - - - - - - - - - -

tab  ( f1,  -10,  10 ) ;

tab  ( fact,  0, 10 ) ;

- - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
void  tab  ( double (* f )  (int ) ,  int  j , int  i )  {
        for ( ; j < = i ;  j ++)
                printf  ( " % d    % f  \ n " , j , ( * f ) ( j )  ) ;
}

double  f1 (int  x ) {
        return  2 *  3.14  * x  ;
}

double  fact (int  x ) {
        double  f = 1 ;  int  i ;
        for ( i = 1 ; i < = x ;  i ++)
                        f * =  i ;
        return  f  ;
}
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
tab  ( f1,  -10,  10 ) ;

tab  ( fact,  0, 10 ) ;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
(DEFUN   tab1  ( f   j   i )
        ( PRINT  ( LIST  j   (FUNCALL  f   j  ) ) )
        ( COND  ( ( =  j   i  )  NIL  )
                ( T  ( tab1  f  ( +   j   1 )  i  ) ) ) )

(DEFUN   tab   ( f   j  i )
        ( COND  ( ( >   j   i  )  NIL  )
                ( T  ( tab1  f    j  i  ) ) ) )

(DEFUN   f1  (x)
        ( *   2    3.14   x  ) )

(DEFUN   f act  (x)
        ( COND   ( ( ZEROP  x )  1.0 )
                ( T  ( *  (FLOAT  x )   ( fact ( -   x    1 ) ) ) ) ) )

( tab  ' f1  -10  10 )

( tab  ' fact   0  10 )
```

Primul limbaj de programare în care s-a introdus o astfel de facilitate a fost Fortran. Reamintim că în acest limbaj unica modalitate de transmitere de parametri este prin adresă, modalitate care poate fi uşor adaptată pentru a transmite şi subprograme (subrutine şi funcţii): adresa corespunzătoare argumentului este adresa din codul obiect de la care încep instrucţiunile subprogramului (aşa numitul punct de intrare în subprogram).

Dacă în lista de parametri formali a unui subprogram Fortran apare numele altui subprogram, atunci parametrul efectiv (argumentul) corespunzător trebuie să fie un nume de subprogram, declarat ca *extern* în programul apelant.

Exemplu: Reluarea exempului anterior, rezolvat de data aceasta în Fortran

```
      SUBROUTINE TAB(F,J,I)
      REAL F
      INTEGER J,I,A
      DO 1 A=J,I
      WRITE(*,2) A,F(A)
2     FORMAT(5X,I4,F10.3)
1     CONTINUE
      RETURN
      END

      REAL FUNCTION F1(X)
      INTEGER X
      F1=2*3.14*X
      RETURN
      END

      REAL FUNCTION FACT(X)
      INTEGER X,I
      REAL F
      F=1.
      DO 1 I=1,X
      F=F*I
1     CONTINUE
      FACT=F
      RETURN
      END

C     PROGRAMUL PRINCIPAL
      EXTERNAL F1,FACT
      REAL F1,FACT
      -------
      CALL TAB(F1,-10,10)
      CALL TAB(FACT,0,10)
      -------
      STOP
      END
```

♦ SUBPROGRAME GENERICE .

```
generic
        type tip_el is private ;
        type vec is array ( integer range < > ) of tip_el ;
        zero : tip_el ;
        with function " + " ( x , y : tip_el ) return tip_el ;

function aplica ( v : vec ) return tip_el ;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

function aplica ( v : vec ) return tip_el is
        rez : tip_el : = zero ;

begin
        for i in v ' first . . v ' last loop
                rez : = rez + v ( i ) ;
        end loop ;
        return rez ;

end aplica ;



type v_int is array ( integer range < > ) of integer ;

type v_real is array ( integer range < > ) of real ;

function sum is new aplica ( integer, v_int, 0, " + " ) ;

function prod is new aplica ( real, v_real, 1.0, " * " ) ;



function ad_inv ( x , y : integer ) return integer is
begin
        if y = 0 then
                return 0 ;
        else
                return x + 1 / y ;
        end if
end ad_inv ;
- - - - - - - - - - - - - - - - - - - - - - -
function s_inv is new aplica ( integer, v_int, 0 , ad_inv ) ;
```

FUNCȚII GENERICE ÎN C++

```
template <class T> void sortare(T  *tablou, int dim)
……..
void main(void)
   {
        int tablouintregi[10]={…};
        double tabloureale[20]={…};
       ……
        sortare(tablouintregi, 10);
        sortare(tabloureale, 20);
        …….
   }


template <class T> void sortare(T  *tablou, int dim)
   {
      register int i, j;
      T temp;
      for (i=1; i<dim; i++)
         for (j=dim-1; j>=i; j--)
            if (tablou[j-1] > tablou[j])
               {
                  temp = tablou[j-1];
                  tablou[j-1] = tablou[j];
                  tablou[j] = temp;
               }
   }
```